IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: **DeWitt, Jr. et al.**	§	
	§	Group Art Unit: **2181**
Serial No. **10/757,269**	§	
	§	Examiner: **Lai, Vincent**
Filed: **January 14, 2004**	§	
	§	
For: **Method and Apparatus for Autonomically Initiating Measurement of Secondary Metrics Based on Hardware Counter Values for Primary Metrics**	§	

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

35525
PATENT TRADEMARK OFFICE
CUSTOMER NUMBER

APPEAL BRIEF (37 C.F.R. 41.37)

This brief is in furtherance of the Reinstatement of Appeal, filed in this case on September 11, 2007.

No fees are believed to be required. If, however, any fees are required, I authorize the Commissioner to charge these fees which may be required to IBM Corporation Deposit Account No. 09-0447. No extension of time is believed to be necessary. If, however, an extension of time is required, the extension is requested, and I authorize the Commissioner to charge any fees for this extension to IBM Corporation Deposit Account No. 09-0447.

REAL PARTY IN INTEREST

The real party in interest in this appeal is the following party: International Business Machines Corporation of Armonk, New York.

RELATED APPEALS AND INTERFERENCES

With respect to other appeals or interferences that will directly affect, or be directly affected by, or have a bearing on the Board's decision in the pending appeal, there are no such appeals or interferences.

STATUS OF CLAIMS

A. TOTAL NUMBER OF CLAIMS IN APPLICATION

Claims in the application are: 1-29

B. STATUS OF ALL THE CLAIMS IN APPLICATION

1. Claims canceled: none
2. Claims withdrawn from consideration but not canceled: none
3. Claims pending: 1-29
4. Claims allowed: none
5. Claims rejected: 1-29
6. Claims objected to: none

C. CLAIMS ON APPEAL

The claims on appeal are: 1-29

STATUS OF AMENDMENTS

No amendment after final rejection was filed for this case.

SUMMARY OF CLAIMED SUBJECT MATTER

A. CLAIM 1 - INDEPENDENT

The present invention provides a method in a data processing system for processing instructions. (Specification, page 87, line 2, to page 90, line 17) The present invention determines whether an indicator is associated with the instruction in response to receiving an instruction at a processor in the data processing system, wherein the indicator identifies the instruction as one that is to be monitored by a performance monitor unit. (Specification, page 90, lines 22-20; and page 66, line 21, to page 67, line 8) The present invention enables counting, by the processor, of each first event associated with a primary metric of the execution of the instruction if the indicator is associated with the instruction, wherein the processor autonomically increments the count of the first events associated with the primary metric of the execution of the instruction in a first hardware counter. (Specification, page 67, line 12, to page 68, line 18) The present invention determines if the count of the first events associated with the primary metric of the execution of the instruction stored in the first hardware counter satisfies a predetermined relationship with a threshold value. (Specification, page 91, lines 1-19) The present invention enables counting, by the processor, of each second event associated with a secondary metric of the execution of a portion of code associated with the instruction, wherein the processor autonomically increments the count of the second events associated with the secondary metric of the execution of a portion of code associated with the instruction in a second hardware counter. (Specification, page 91, line 19, to page 92, line 9)

B. CLAIM 15 - INDEPENDENT

The present invention provides for a computer program product in a recordable-type computer readable medium for processing instructions. (Specification, page 125, line 23, to page 126, line 11; and Specification, page 87, line 2, to page 90, line 17) The present invention provides first instructions for determining whether an indicator is associated with the instruction in response to receiving an instruction at a processor in the data processing system, wherein the indicator identifies the instruction as one that is to be monitored by a performance monitor unit. (Specification, page 90, lines 22-20; and page 66, line 21, to page 67, line 8) The present invention provides second instructions for enabling counting, by the processor, of each first event

associated with a primary metric of the execution of the instruction if the indicator is associated with the instruction, wherein the processor autonomically increments the count of the first events associated with the primary metric of the execution of the instruction in a first hardware counter. (Specification, page 67, line 12, to page 68, line 18) The present invention provides third instructions for determining if the count of the first events associated with the primary metric of the execution of the instruction stored in the first hardware counter satisfies a predetermined relationship with a threshold value. (Specification, page 91, lines 1-19) The present invention provides fourth instructions for enabling counting, by the processor, of each second event associated with a secondary metric of the execution of a portion of code associated with the instruction, wherein the processor autonomically increments the count of the second events associated with the secondary metric of the execution of a portion of code associated with the instruction in a second hardware counter. (Specification, page 91, line 19, to page 92, line 9)

A person having ordinary skill in the art would be able to derive computer instructions on a computer readable medium as recited in claim 15, as well as dependent claims 16-28, given **Figures 29** and **39**, performing the steps described in the specification at page 66, line 21, to page 68, line 19; and page 90, line 18, to page 92, line 9, without undue experimentation.

C. CLAIM 29 - INDEPENDENT

The present invention provides for an apparatus for processing instructions. (Specification, page 23, lines 12-21; page 26, lines 10-13; page 28, line 19, to page 29, line 9; and Specification, page 87, line 2, to page 90, line 17) The present invention provides means for determining whether an indicator is associated with the instruction in response to receiving an instruction at a processor in the data processing system, wherein the indicator identifies the instruction as one that is to be monitored by a performance monitor unit. (Specification, page 90, lines 22-20; and page 66, line 21, to page 67, line 8) The present invention provides means for enabling counting, by the processor, of each first event associated with a primary metric of the execution of the instruction if the indicator is associated with the instruction, wherein the processor autonomically increments the count of the first events associated with the primary metric of the execution of the instruction in a first hardware counter. (Specification, page 67, line 12, to page 68, line 18) The present invention provides means for determining if the count of the

first events associated with the primary metric of the execution of the instruction stored in the first hardware counter satisfies a predetermined relationship with a threshold value. (Specification, page 91, lines 1-19) The present invention provides means for enabling counting, by the processor, of each second event associated with a secondary metric of the execution of a portion of code associated with the instruction, wherein the processor autonomically increments the count of the second events associated with the secondary metric of the execution of a portion of code associated with the instruction in a second hardware counter. (Specification, page 91, line 19, to page 92, line 9)

The apparatus recited in claim 29 may be an apparatus comprised of functional units such as execution units **220**, **222**, **224**, **226**, **228**, and **230** in processor **210** of **Figure 2**. The processor and functional units providing means for determining and means for enabling to perform the steps described in the specification at page 66, line 21, to page 68, line 19; and page 90, line 18, to page 92, line 9, without undue experimentation.

GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

The grounds of rejection to review on appeal are as follows:

1. Whether claims 1-5, 7-9, 14-19, 21-23, 28 and 29 are anticipated by Holmberg (U.S. Patent # 6,233,679 B1) under 35 U.S.C. § 102(b); and

2. Whether claims 6, 10-13, 20 and 24-27 are obvious over Holmberg (U.S. Patent # 6,233,679 B1) in view of Yates et al (U.S. Patent # 6,549,959) under 35 U.S.C. § 103.

ARGUMENT

A. GROUND OF REJECTION 1 (Claims 1-5, 7-9, 14-19, 21-23, 28 and 29)

Claims 1-5, 7-9, 14-19, 21-23, 28 and 29 stand rejected under 35 U.S.C. § 102(b) as being anticipated by Holmberg (U.S. Patent # 6,233,679 B1).

The teachings of the cited Holmberg reference and the features of Claim 1 are fundamentally different. The teachings of Holmberg are specifically directed to techniques used to predict branches to determine whether it is more or less likely that a branch instruction that is subsequently encountered will take the branch or not (col. 2, lines 16-39; col. 5, lines 35-43). As a part of such branch prediction technique, a background program is executed to scan memory in order to locate conditional branch instructions that are resident in memory (col. 2, lines 30-33). Importantly, this process is *not part of actual execution of the conditional branch instruction* by a processor, but instead is a pre-execution processing stage *to locate such conditional branch instructions* in memory so that a prediction can be made, before the instruction is actually executed by a processor, as to whether the branch will be taken – and such *pre-execution processing of conditional branch instructions* are a fundamental premise of branch prediction techniques.

In contrast, per the features of the appealed claims, a performance monitoring technique is provided where performance of actual machine execution occurs to advantageously allow for identification of programming hot spots and other performance related issues. Instructions are received at a processor in the data processing system. If a selected indicator is associated with the instruction received at the processor, *counting of each event associated with the **execution of** the instruction is enabled.* In addition, functionality is provided in the performance monitoring application for initiating the measurement of *secondary metrics* with regard to identified instructions, such as data addresses, ranges of identified instructions, or ranges of identified data addresses, based on counter values for *primary metrics*. Thus, for example, when a primary metric counter meets or exceeds a predetermined threshold value, an interrupt may be generated. In response to receiving the interrupt, counters associated with the measuring of secondary metrics of a range of instructions/data addresses may be initiated. In this way, areas of particular interest may first be identified using the primary metric performance counters with more detailed

information being obtained through the use of secondary metric performance counters directed to measuring metrics associated with the particular area of interest.

For a prior art reference to anticipate in terms of 35 U.S.C. 102, every element of the claimed invention must be identically shown in a single reference. *In re Bond*, 910 F.2d 831, 15 USPQ2d 1566 (Fed. Cir. 1990). Appellants will now show that every element recited in these Claims 1-5, 7-9, 14-19, 21-23, 28 and 29 is not identically shown in a single reference, and thus such claims have been erroneously rejected under 35 U.S.C. § 102(b).

A.1. Claims 1, 2, 9, 15, 16, 23, 29

Specifically with respect to Claim 1, such claim recites:

> A method in a data processing system for processing instructions, the method comprising:
>
> *responsive to receiving an instruction at a processor* in the data processing system, determining whether an indicator is associated with the instruction, *wherein the indicator identifies the instruction as one that is to be monitored by a performance monitor unit*;
>
> enabling *counting*, by the processor, *of each first event associated with a primary metric of the execution of the instruction* if the indicator is associated with the instruction, wherein *the processor autonomically increments the count* of the first events associated with the primary metric of the execution of the instruction in a first hardware counter;
>
> determining if the count of the first events associated with the primary metric of the execution of the instruction stored in the first hardware counter satisfies a predetermined relationship with a threshold value; and
>
> enabling *counting*, by the processor, of *each second event associated with a secondary metric of the execution of a portion of code associated with the instruction*, wherein *the processor autonomically increments the count* of the second events associated with the secondary metric of the execution of a portion of code associated with the instruction in a second hardware counter. (emphasis added)

As can be seen, the processor plays a crucial role with respect to the features of Claim 1. First, and a catalyst the for entire performance monitoring process, an instruction is received at this processor, and in response thereto, a determination is made as to whether there is an indicator associated with the instruction – with such indicator indicating that this instruction received at the processor is one that is to be monitored by a performance execution unit. If this indicator indicates that such monitoring it to occur, this same processor (where the instruction

was received at) enables *counting of events associated with a metric of actual execution of this instruction.* As a part of such counter enabling by the processor (where the instruction was received at), this same processor autonomically increments the count of these events (i.e. a plurality of events) that are associated with *actual execution of the instruction.* Further yet, this same processor (where the instruction was received at) enables *counting of second events* (i.e. a plurality of events) *associated with a secondary metric of actual execution of a portion of code associated with this instruction* – which is thus counting events of *different code* ('a portion of *code* association with the instruction, as contrasted to the *instruction* itself). Appellants urge that the cited reference does not teach such a processor, as will now be shown in detail.

In rejecting Claim 1, the Examiner cites Holmberg's passage at column 3, lines 46-48 as teaching the step of receiving an instruction at a processor. Appellants urge that this cited passage describes a memory interface block 107 that is used for fetching instructions from memory which are then provided to instruction decode block 109. Thus, according to the Examiner, either block 107 or block 109 are equivalent to the claimed 'processor' as this passage is being cited as teaching an instruction being received by a 'processor'. However, neither of these blocks 107 nor 109 *enable counting or increment a count of events* as required by the 'processor' recited in Claim 1. Instead, and as described by Holmberg at col. 5, lines 8-14, the counters are controlled by the background program that is executing. This background program is not executed by blocks 107 and 109, and thus blocks 107 and 109 – one of which is alleged to be equivalent to the claimed 'processor' – do not enable counters or increment counts, as claimed. Thus, the cited reference does not teach the claimed 'processor' that performs each of the numerous enumerated functions identified above, as the cited reference does not teach counter enabling or count incrementing being performed by the same processor for which an instruction was received at. Thus, for this reason alone, it is urged that Claim 1 is not anticipated by the cited reference.

Still further, per the features of Claim 1 a determination is made as to whether an indicator is associated with the instruction received at the processor, where such indicator indicates that the instruction received at the processor is one that is to be monitored. In rejecting this 'monitoring indicator' aspect of Claim 1, the Examiner cites Holmberg's description at col. 5, lines 14-21 as teaching the claimed feature that 'counters count when encountering a type of branch instruction and thus must be able to identify and determine when such events occur'.

Appellants respectfully submit that the fact that counters count when encountering a type of branch does not teach or otherwise suggest the claimed monitoring indicator associated with an instruction received by the processor, where such monitoring indicator *identifies the instruction as one that is **to be** monitored by a performance monitor unit* – i.e., monitored at a later or future point in time. The Holmberg counters count actual branch instructions currently being executed (i.e. a current event, as described by Holmberg at col. 5, line 1), and the number of times the conditional branch was taken (i.e., past events as described by Holmberg at col. 4, lines 64-67). These counters, or the fact that they count, does not teach or otherwise suggest the claimed monitoring indicator that *identifies the instruction as one that is **to be** monitored by a performance monitor unit* – i.e. an instruction that is to be monitored at a later or future point in time, as claimed. Thus, it is further shown that Claim 1 has been erroneously rejected, as such reference does not teach the claimed 'indicator' having all of the particular properties associated with it, as defined in the claim and described in detail hereinabove.

The criticality of this missing claimed monitoring indicator feature is particularly evident in that, per the features of Claim 1, the *counting is enabled if the indicator* is found to be in association with the instruction received by the processor. Fromberg's counting by counters, alleged to teach the claimed monitoring indicator feature of Claim 1, are not enabled upon determining the existence of the counters or their counting, which would have to occur in order for such counters to read upon the claimed feature of "*enabling counting*, by the processor, *of* each first event associated with a primary metric of the execution of the instruction *if the indicator is associated with the instruction*" as the Examiner states that the 'indicator' is Holmberg's counting by the counters. This leads to the illogical result that Holmberg enables counting by the counters if the counters are determined to be counting – an awkward, and quite simply illogical, result. Thus, it is further shown – when viewed in light of the counter enabling aspect of Claim 1, where the counter is enabled based on the monitoring indicator – that contrary to the Examiner's assertion, the Holmberg counter counting does not teach or otherwise suggest the claimed 'indicator' having all of the particular properties associated with it, as defined in the claim and described in detail hereinabove.

Still further, per the features of Claim 1 there are two different – and fundamentally different – items that are being counted by the counters. The first items that are counted are

(first) events associated with an execution metric for the *actual instruction itself* (the instruction being the instruction received at the processor for which a monitoring indicator is associated with), whereas the second items that are counted are an execution metric of *a portion of code associated with the instruction* – which is different code than the actual instruction. In contrast, per the teachings of Holmberg, the two counters count the number of times *a given conditional instruction* was executed (col. 5, line 1), and the number of times a branch was taken when *this same given conditional instruction* was executed (col. 4, lines 64-67). Thus, Holmberg's counters are both counting events associated with execution of the same conditional instruction. Restated, per Claim 1 there are two completely *different codes* being monitored/counted – the actual *instruction* itself as well as *code* associated with the instruction – whereas per the teachings of Holmberg two different aspects of the *same instruction* are counted. Thus, it is further urged that Claim 1 is not anticipated by the cited reference as there are additional claimed features that are not identically shown in a single reference.

Counting of such *secondary events for code other than the instruction itself* advantageously provides functionality in the performance monitoring application for initiating the measurement of *secondary metrics* with regard to identified instructions, such as data addresses, ranges of identified instructions, or ranges of identified data addresses, based on counter values for *primary metrics*. Thus, for example, when a primary metric counter meets or exceeds a predetermined threshold value, an interrupt may be generated. In response to receiving the interrupt, counters associated with the measuring of secondary metrics of a range of instructions/data addresses may be initiated. In this way, areas of particular interest may first be identified using the primary metric performance counters with more detailed information being obtained through the use of secondary metric performance counters directed to measuring metrics associated with the particular area of interest.

Thus, it is urged that Claim 1 (and Claims 2, 9, 15, 16, 23 and 29) has been erroneously rejected under 35 U.S.C. 102(b), as every element of the claimed invention is not identically shown in a single reference.

A.2. Claims 3 and 17

Appellants initially urge error in the rejection of Claim 3 (and similarly for Claim 17) for the same reasons as those given above with respect to Claim 1 (from which Claim 3 depends upon).

Still further, Claim 3 refines characteristics in the monitoring indicator, where (i) such indicator is stored in a shadow cache, and (2) the determination as to whether such indicator is associated with the instruction received at the processor is done by the processor (for which the instruction was received at) checking such shadow cache. In rejecting Claim 3, the Examiner states that Holmberg teaches all features of Claim 3 since 'The MAR is the shadow cache' 'counters count when encountering a type of branch instruction'. Appellants respectfully submit that no Holmberg processor checks the MAR – which is alleged to be equivalent to the claimed 'shadow cache'. Claim 3 expressly states "wherein the processor checks the performance instrumentation shadow cache", so if Holmberg's MAR is equivalent to such shadow cache, the Holmberg reference would have to teach a processor checking the MAR in order to anticipate the features of Claim 3. However, Holmberg does not teach a processor checking such MAR – which is a register containing the address of a conditional branch. This MAR, to the extent it is checked at all, is read by a hardware comparator 203 (see Holmberg's Figure 2, elements 201 and 203 as further described by Holmberg at col. 4, lines 47-55). This hardware comparator cannot be reasonably construed to be equivalent to the claimed 'processor' as the actual *instructions* themselves are received at the claimed processor, whereas Holmberg's comparator (which is the only element that checks the MAR (and such MAR is alleged to be equivalent to the claimed shadow cache)) merely receives *addresses* (Holmberg Figure 2, elements 201, 203 and 205 as further described by Holmberg at col. 4, lines 47-63). Thus, Holmberg does not teach a processor - for which an instruction is received at, and which enables counters and increments counts – that checks a shadow register: (i) either as per the particular features of Claim 3 in order to determine whether a monitoring indicator is associated with the instruction received at such processor, or (ii) for any other reason. Thus, it is further shown that Claim 3 (and Claim 17) has been erroneously rejected as there are additional claimed features (shadow cache checking by a processor to determine monitoring indicator) that are not taught by the cited reference.

A.3. Claims 4 and 18

Appellants initially urge error in the rejection of Claim 4 (and similarly for Claim 18) for the same reasons as those given above with respect to Claim 1 (from which Claim 4 depends upon).

Still further, Claim 4 refines where the monitoring indicator is maintained – a spare bit in a field in a bundle for which the instruction was received in by an instruction cache. The Examiner states that such bundle field spare bit is taught by Holmberg at Figure 1, col. 3, lines 63-64 and col. 5, lines 3-7 in that there Holmberg teaches a branch prediction bit and a parameter that indicates where a branch may occur. Appellants urge error, as the Holmberg branch prediction bit is substantially different from the claimed monitoring indicator. For example, Holmberg's branch predication bit indicates whether Holmberg's branch prediction logic has determined whether it is *more or less likely that a branch will be taken* (Holmberg col. 5, lines 35-43). In contrast, the claimed monitoring indicator *identifies the instruction as one that is to be monitored by a performance monitor unit* by enabling counters, etc.

The difference between the Holmberg branch prediction bit and the claimed monitoring indicator can also be seen in that Holmberg's *counters have already been enabled and counted* as a part of the process used to determine whether or not to set the Holmberg branch prediction bit (see Holmberg flowchart at Figure 3a, as further described by Holmberg at col. 5, lines 3-51), whereas the claimed monitoring indicator is used to actually *enable the counters* ("*enabling counting*, by the processor, of each first event associated with a primary metric of the execution of the instruction *if the indicator is associated with the instruction*"). Quite simply, Holmberg's branch prediction bit cannot reasonably be construed to be the claimed monitoring indicator, as counters are not enabled if such branch prediction bit is determined to be associated with an instruction – and in fact the Holmberg *counters must have already been previously enabled* in order to gather system statistics to determine whether or not to set such branch prediction bit (Holmberg col. 2, lines 34-39). Thus it has been further shown that Holmberg's branch prediction bit is not equivalent to the claimed monitoring indicator, as alleged by the Examiner in rejecting Claim 4, and therefore Claim 4 (and Claim 18) has been erroneously rejected.

A.4. Claims 5 and 19

Appellants initially urge error in the rejection of Claim 5 (and similarly for Claim 19) for the same reasons as those given above with respect to Claim 1 (from which Claim 5 depends upon).

Further, Claim 5 recites that the indicator is a *separate instruction* (which is therefore different from the *instruction* received at the processor). In rejecting Claim 5, the Examiner states this is taught by Holmberg since Holmberg teaches the indicator as being a specific type of branch instruction. Restated, per the Examiner's interpretation *the instruction itself* for which all the processing is associated with (for performing the branch prediction technique) is the indicator. In contrast, and per the plain meaning of Claim 5, a *separate instruction* (other than the *instruction* for which performance monitoring/processing is associated with) is the indicator. Thus, it is shown that Holmberg's branch instruction *itself* as being the indicator (for which all the associated processing occurs in response thereto) does not teach a *separate* instruction being the indicator (for which all the associated processing occurs in response thereto). Therefore, there are additional claimed features not taught by the cited reference, further evidencing that Claim 5 (and Claim 19) has been erroneously rejected.

A.5. Claims 7 and 21

Appellants initially urge error in the rejection of Claim 7 (and similarly for Claim 21) for the same reasons as those given above with respect to Claim 1 (from which Claim 7 depends upon).

In addition, Claim 7 recites further aspects of how the indicator determination is performed, and in particular states that the instruction cache makes such indicator determination. In rejecting Claim 7, the Examiner states that Holmberg teaches all features of Claim 7 at col. 5, lines 14-21 since 'counters' count when encountering a type of branch instruction and thus must be able to identify and determine when such events occurs'. Appellants urge error in such analysis. During examination, the claims must be interpreted as broadly as their terms reasonably allow. *In re American Academy of Science Tech Center*, 367 F.3d 1359, 1369, 70 USPQ2d 1827, 1834 (Fed. Cir. 2004). This means that the words of the claim must be given their **plain meaning** unless the plain meaning is inconsistent with the specification. *In re Zletz*,

893 F.2d 319, 321, 13 USPQ2d 1320, 1322 (Fed. Cir. 1989) (discussed below); *Chef America, Inc. v. Lamb-Weston, Inc.*, 358 F.3d 1371, 1372, 69 USPQ2d 1857 (Fed. Cir. 2004) (Ordinary, simple English words whose meaning is clear and unquestionable, absent any indication that their use in a particular context changes their meaning, are **construed to mean exactly what they say**. Thus, "heating the resulting batter-coated dough to a temperature in the range of about 400°F to 850°F" required heating the dough, rather than the air inside an oven, to the specified temperature.) MPEP 2111.01(I) (emphasis added by Appellants). It is urged that the Examiner is not interpreting the claim terms in accordance with their normal, plain meaning as the Holmberg *counters* are clearly not equivalent to the claimed *instruction cache* as those terms are commonly known to mean to persons of ordinary skill in the art. Thus, anything that the Holmberg *counters* may determine does not teach the claimed feature of "determining, *by an instruction cache*, whether the indicator is present in a field within the instruction" (emphasis added by Appellants). Thus, it is further shown that Claim 7 (and Claim 21) has been erroneously rejected, as there are additional claimed features that are not identically shown in a single reference.

A.6. Claims 8 and 22

Appellants initially urge error in the rejection of Claim 8 (and similarly for Claim 22) for the same reasons as those given above with respect to Claim 1 (from which Claim 8 depends upon).

Still further, Claim 8 explicitly recites "wherein the enabling the counting of first events includes sending a first signal to the performance monitor unit, wherein the performance monitor unit counts each first event associated with execution of the instruction using the first hardware counter, and wherein enabling the counting of second events includes sending a second signal to the performance monitor unit, wherein the performance monitor unit counts each second event associated with execution of a portion of code associated with the instruction using the second hardware counter". As can be seen, Claim 8 further refines the enablement of the two counters where two different signals (a first signal and a second signal) are sent to the performance monitoring unit for enabling counting by the first and second counters, respectively. In rejecting Claim 8, the Examiner states the sending of two signals to a performance monitoring unit to enable counting of the respective first and second counters is taught by Holmberg at col. 4, line

47 – col. 5, line 2 since 'Two types of statistics are counted and recorded'. Appellants urge clear error, as the fact that two *types* of statistics are counted/recorded by Holmberg does not establish any teaching associated with *how the counters are actually enabled*, either as per the particular first and second signal features recited in Claim 8. Thus, it is further shown that Claim 8 (and Claim 22) has been erroneously rejected, as there are additional claimed features that are not identically shown in a single reference.

B. GROUND OF REJECTION 2 (Claims 6, 10-13, 20 and 24-27)

Claims 6, 10-13, 20 and 24-27 stand rejected under 35 U.S.C. § 103 as being obvious over Holmberg (U.S. Patent # 6,233,679 B1) in view of Yates et al (U.S. Patent # 6,549,959).

B.1. Claims 6 and 20

Appellants urge error in the rejection of Claim 6 (and similarly for Claim 20) for similar reasons to those given above with respect to Claim 1, as the newly cited reference to Yates does not overcome the teaching deficiencies identified above with respect to Claim 1.

B.2. Claims 10 and 24

Appellants initially urge error in the rejection of Claim 10 (and similarly for Claim 24) for similar reasons to those given above with respect to Claim 1, as the newly cited reference to Yates does not overcome the teaching deficiencies identified above with respect to Claim 1.

Further, Claim 10 recites "wherein *enabling counting*, by the processor, of each second event associated with the secondary metric of the execution of the portion of code associated with the instruction *includes: generating an interrupt in response to a determination that the count of the first events meets or excess the threshold value*; and sending the interrupt to an interrupt handler of a performance monitoring application, wherein *the interrupt handler of the performance monitoring application initiates counting of each second event* associated with a secondary metric of the execution of a portion of code associated with the instruction". As can be seen, these claimed features are further directed to the ability of the performance monitoring application to advantageously initiate the measurement of *secondary metrics* with regard to identified instructions, such as data addresses, ranges of identified instructions, or ranges of identified data addresses, based on counter values for *primary metrics*. Thus, for example, when

a primary metric counter meets or exceeds a predetermined threshold value, an interrupt may be generated. In response to receiving the interrupt by the interrupt handler, counters associated with the measuring of secondary metrics of a range of instructions/data addresses may be initiated. In this way, areas of particular interest may first be identified using the primary metric performance counters with more detailed information being obtained through the use of secondary metric performance counters directed to measuring metrics associated with the particular area of interest.

In rejecting Claim 10, the Examiner acknowledges that the cited Holmberg reference does not teach interrupts, but states that 'Yates teaches generating an interrupt in response to a determination that the count of the first events meets or excess the threshold value (See column 55, lines 58-60)'. There, Yates states:

> "Hot spot detector 122 conveys this information to TAXi translator 124, which in turn translates the binary"

Appellants respectfully submit that this cited Yates passage does not describe any type of interrupt generation, either in response to a determination that the count of the first events meets or excess the threshold value, as claimed. Thus, it is urged that the Examiner has failed to properly establish a prima facie showing of obviousness with respect to Claim 10, and therefore Claim 10 has been erroneously rejected[1].

Still further regarding Claim 10, the claimed interrupt generation step depends upon the claimed step of enabling counting of second events. The Examiner expressly acknowledges on page 9 of the present Office Action dated June 12, 2007 that the cited Yates reference does not teach a second counter thus no secondary metric, or counting of a second event. If Yates does not teach a second counter or the counting of a second event, then it is impossible for Yates to teach a sub-step (generating an interrupt) that is performed as a part of the claimed step of *enabling counting*, by the processor, of each *second event* associated with the *secondary metric*.

---

[1] To establish prima facie obviousness of a claimed invention, all of the claim limitations must be taught or suggested by the prior art. MPEP 2143.03. *See also, In re Royka*, 490 F.2d 580 (C.C.P.A. 1974). If the examiner fails to establish a prima facie case, the rejection is improper and will be overturned. *In re Fine*, 837 F.2d 1071, 1074, 5 USPQ2d 1596, 1598 (Fed. Cir. 1988).

Still further with respect to Claim 10, the Examiner cites Yates at col. 5 lines 11-13 as teaching sending the generated interrupt to an interrupt handler of the performance monitoring application. There, Yates states:

> "There may be a defined mapping between resources of the first architecture and resources of the second, the mapping assigning corresponding resources of the two architectures to a common physical resource of a computer when the resources serve analogous functions in the calling conventions of the two architectures."

As can be seen, this cited Yates passage describes resource mapping between two architectures, and has no bearing on any type of interrupt handler such as the claimed interrupt handler of a performance monitoring application. Thus, it is further urged that a proper prima facie showing of obviousness has not been established with respect to Claim 10, and accordingly it is further urged that Claim 10 (and Claim 24) has been erroneously rejected.

Still further, the claimed interrupt sending step depends upon the step of enabling counting of second events. For similar reasons to those described above, since Yates does not enable the counting of second events it cannot teach a sub-step that is performed to achieve the enabling the counting of such (missing) second events. Thus, it is further urged that a proper prima facie showing of obviousness has not been established with respect to Claim 10, and accordingly it is further urged that Claim 10 (and Claim 24) has been erroneously rejected.

Still further with respect to Claim 10, the Examiner cites Yates teaching at col. 67 lines 36-45 as teaching a performance monitor application interrupt handler that initiates counting of events, in that 'The abort will affect the profiler'. Appellants urge that there, Yates states:

> "A transition from X86 code to Tapestry code (for instance, a successful probe exception, see section VI, infra) may be an abort 550, 552 event. Profiler 400 is configured to allow the choice between entirely discarding the aborted packet or padding out and then spilling the partial packet to the ring buffer before abort 550, 552 occurs. This choice is implemented in the code of the X86-to-Tapestry transition handler 320. FIG. 5b is a block diagram of a portion of profiler 400, the logic 554 to collect and format a profile entry 430, 440 into a processor register."

As can be seen, this cited passage describes either: (i) entirely discarding an aborted packet, or (ii) padding out and then spilling the partial packet to a ring buffer before an abort occurs. This

cited passage does not describe any type event counting initiation by an interrupt handler of a performance monitor application, as expressly recited in Claim 10. Thus, it is further urged that a proper prima facie showing of obviousness has not been established with respect to Claim 10 (and Claim 24), and accordingly it is further urged that Claim 10 (and Claim 24) has been erroneously rejected.

B.3. Claims 11 and 25

Appellants initially urge error in the rejection of Claim 11 (and similarly for Claim 25) for the same reasons as those given above with respect to Claim 10 (from which Claim 11 depends upon).

Still further with respect to Claim 11, such claim recites "wherein the interrupt handler *instruments other instructions* in the portion of code associated with the instruction *to include the indicator*". As can be seen, further aspects of the claimed interrupt handler are claimed, including instrumenting other instructions in the portion of code (for which second events are counted) to include the monitoring indicator. In rejecting such code instrumentation feature, the Examiner states that Yates teaches this feature at col. 55, lines 58-60. There, Yates states:

> "Hot spot detector 122 conveys this information to TAXi translator 124, which in turn translates the binary"

Appellants respectfully submit that this cited Yates passage does not describe any type of operations being performed by an interrupt handler, either *instrumenting other instructions* in the portion of code associated with the instruction *to include the monitoring indicator*, or any other type of interrupt handler processing/actions. Thus, it is further urged that a proper prima facie showing of obviousness has not been established with respect to Claim 11 (and Claim 25), and accordingly it is further urged that Claim 11 (and Claim 25) has been erroneously rejected.

B.4. Claims 12 and 26

Appellants initially urge error in the rejection of Claim 12 (and similarly for Claim 26) for the same reasons as those given above with respect to Claim 10 (of which Claim 12 depends upon).

Still further with respect to Claim 12, such claim recites "wherein the interrupt handler initiates the second hardware counter and associates the second hardware counter with the portion of code". As can be seen, further aspects of the claimed interrupt handler are claimed, including initiating the second hardware counter. The Examiner takes the position that this claimed feature would be obvious, alleging that Holmberg teaches a second counter and Yates teaches an interrupt handler. While it may be true that one element (counter) is taught by one reference (Holmberg) and the other element (interrupt handler) is taught by the other reference (Yates), such allegation still does not establish any synergistic co-action(s) that are performed using such two elements in tandem – such as the claimed feature of the interrupt handler *initiating* a second hardware counter. The Examiner has merely established the existence of these two elements, but no co-action therebetween. A person of ordinary skill in the art would not have been motivated to initiate Holmberg's second timer with an interrupt handler, as Holmberg expressly requires that such (second) timer be initiated *by a background routine at the same time* that the first counter is initiated *by the background routine*, as both counters are used to count different aspects of the same variable (the branch instruction). If one timer were allowed to be started asynchronously with respect to the other timer (such as by use of an interrupt handler), the entire premise of Holmberg's desire to count two different aspects with respect to a single instruction would be defeated as the counters would not start at the same time – and thus would not be synchronized with respect to one another in attempting to count two different aspects of the same variable (the branch instruction). As such proposed change to the teachings of the cited references would in effect eviscerate one of the fundamental premises upon which Holmberg is concerned about, as described above, a person of ordinary skill in the art would not have been motivated to make such a change – further evidencing that such claim has been erroneously rejected as there is no motivation to modify such teachings to include the claimed synergistic co-action between the interrupt handler and the second timer. Thus, it is further urged that Claim 12 (and Claim 26) has been erroneously rejected.

Still further with respect to Claim 12, none of the cited references teach or suggest the claimed feature of an interrupt handler that associates a second hardware counter with the portion of code (for which second events are counted), nor has the Examiner alleged any such teaching or suggestion. Thus, it is further urged that Claim 12 (and Claim 26) has been erroneously rejected, as a proper prima facie showing of obviousness has not been established by the Examiner.

B.5. Claims 13 and 27

Appellants urge error in the rejection of Claim 13 (and similarly for Claim 27) for similar reasons to those given above with respect to Claim 1, as the newly cited reference to Yates does not overcome the teaching deficiencies identified above with respect to Claim 1.

Appellants have thus shown numerous and substantial error in the Examiner's final rejection of all pending claims, and accordingly respectfully requests that the Board reverse the final rejection of all such claims.

/Wayne P. Bailey/
Wayne P. Bailey
Reg. No. 34,289
YEE & ASSOCIATES, P.C.
PO Box 802333
Dallas, TX 75380
(972) 385-8777

CLAIMS APPENDIX

The text of the claims involved in the appeal are:

1. A method in a data processing system for processing instructions, the method comprising:

responsive to receiving an instruction at a processor in the data processing system, determining whether an indicator is associated with the instruction, wherein the indicator identifies the instruction as one that is to be monitored by a performance monitor unit;

enabling counting, by the processor, of each first event associated with a primary metric of the execution of the instruction if the indicator is associated with the instruction, wherein the processor autonomically increments the count of the first events associated with the primary metric of the execution of the instruction in a first hardware counter;

determining if the count of the first events associated with the primary metric of the execution of the instruction stored in the first hardware counter satisfies a predetermined relationship with a threshold value; and

enabling counting, by the processor, of each second event associated with a secondary metric of the execution of a portion of code associated with the instruction, wherein the processor autonomically increments the count of the second events associated with the secondary metric of the execution of a portion of code associated with the instruction in a second hardware counter.

2. The method of claim 1, wherein the instruction is received in an instruction cache in the processor.

3. The method of claim 1, wherein the indicator is stored in a performance instrumentation shadow cache and wherein the processor checks the performance instrumentation shadow cache to determine whether the indicator is associated with the instructions.

4. The method of claim 1, wherein the instruction is received in a bundle by an instruction cache in the processor and wherein the indicator comprises at least one spare bit in a field in the bundle.

5. The method of claim 1, wherein the indicator is a separate instruction.

6. The method of claim 1, wherein the first events include at least one of an entry into a module, an exit from a module, an entry into a subroutine, an exit from a subroutine, an entry into a function, an exit from a function, a start of input/output, a completion of input/output, and the execution of the instruction.

7. The method of claim 1, wherein determining whether an indicator is associated with the instruction comprises:

determining, by an instruction cache, whether the indicator is present in a field within the instruction.

8. The method of claim 1, wherein the enabling the counting of first events includes sending a first signal to the performance monitor unit, wherein the performance monitor unit counts each first event associated with execution of the instruction using the first hardware counter, and

wherein enabling the counting of second events includes sending a second signal to the performance monitor unit, wherein the performance monitor unit counts each second event associated with execution of a portion of code associated with the instruction using the second hardware counter.

9. The method of claim 1, wherein the first hardware counter is a combined counter value hardware counter that stores a combined count from a plurality of other hardware counters.

10. The method of claim 1, wherein enabling counting, by the processor, of each second event associated with the secondary metric of the execution of the portion of code associated with the instruction includes:

generating an interrupt in response to a determination that the count of the first events meets or excess the threshold value; and

sending the interrupt to an interrupt handler of a performance monitoring application, wherein the interrupt handler of the performance monitoring application initiates counting of each second event associated with a secondary metric of the execution of a portion of code associated with the instruction.

11. The method of claim 10, wherein the interrupt handler instruments other instructions in the portion of code associated with the instruction to include the indicator.

12. The method of claim 10, wherein the interrupt handler initiates the second hardware counter and associates the second hardware counter with the portion of code.

13. The method of claim 1, wherein the portion of code associated with the instruction includes at least one of instructions of a same class of instructions as the instruction and instructions within a same method or routine as the instruction.

14. The method of claim 1, wherein the first metric is different from the second metric.

15. A computer program product in a recordable-type computer readable medium for processing instructions comprising:

first instructions responsive to receiving an instruction at a processor in the data processing system, determining whether an indicator is associated with the instruction, wherein the indicator identifies the instruction as one that is to be monitored by a performance monitor unit;

second instructions for enabling counting, by the processor, of each first event associated with a primary metric of the execution of the instruction if the indicator is associated with the instruction, wherein the processor autonomically increments the count of the first events associated with the primary metric of the execution of the instruction in a first hardware counter;

third instructions for determining if the count of the first events associated with the primary metric of the execution of the instruction stored in the first hardware counter satisfies a predetermined relationship with a threshold value; and

fourth instructions for enabling counting, by the processor, of each second event associated with a secondary metric of the execution of a portion of code associated with the instruction, wherein the processor autonomically increments the count of the second events

associated with the secondary metric of the execution of a portion of code associated with the instruction in a second hardware counter.

16. The computer program product of claim 15, wherein the instruction is received in an instruction cache in the processor.

17. The computer program product of claim 15, wherein the indicator is stored in a performance instrumentation shadow cache and wherein the processor checks the performance instrumentation shadow cache to determine whether the indicator is associated with the instructions.

18. The computer program product of claim 15, wherein the instruction is received in a bundle by an instruction cache in the processor and wherein the indicator comprises at least one spare bit in a field in the bundle.

19. The computer program product of claim 15, wherein the indicator is a separate instruction.

20. The computer program product of claim 15, wherein the first events include at least one of an entry into a module, an exit from a module, an entry into a subroutine, an exit from a subroutine, an entry into a function, an exit from a function, a start of input/output, a completion of input/output, and the execution of the instruction.

21. The computer program product of claim 15, wherein the first instructions include:

instructions for determining, by an instruction cache, whether the indicator is present in a field within the instruction.

22. The computer program product of claim 15, wherein the second instructions for enabling the counting of first events include instructions for sending a first signal to the performance monitor unit, wherein the performance monitor unit counts each first event associated with execution of the instruction using the first hardware counter, and wherein enabling the counting of second events includes sending a second signal to the performance monitor unit, wherein the performance monitor unit counts each second event associated with execution of a portion of code associated with the instruction using the second hardware counter.

23. The computer program product of claim 15, wherein the first hardware counter is a combined counter value hardware counter that stores a combined count from a plurality of other hardware counters.

24. The computer program product of claim 15, wherein the fourth instructions for enabling counting, by the processor, of each second event associated with the secondary metric of the execution of the portion of code associated with the instruction include:

instructions for generating an interrupt in response to a determination that the count of the first events meets or excess the threshold value; and

instructions for sending the interrupt to an interrupt handler of a performance monitoring application, wherein the interrupt handler of the performance monitoring application initiates

counting of each second event associated with a secondary metric of the execution of a portion of code associated with the instruction.

25. The computer program product of claim 24, wherein the interrupt handler instruments other instructions in the portion of code associated with the instruction to include the indicator.

26. The computer program product of claim 24, wherein the interrupt handler initiates the second hardware counter and associates the second hardware counter with the portion of code.

27. The computer program product of claim 15, wherein the portion of code associated with the instruction includes at least one of instructions of a same class of instructions as the instruction and instructions within a same method or routine as the instruction.

28. The computer program product of claim 15, wherein the first metric is different from the second metric.

29. An apparatus for processing instructions, the apparatus comprising:

means for determining whether an indicator is associated with the instruction in response to receiving an instruction at a processor in the data processing system, wherein the indicator identifies the instruction as one that is to be monitored by a performance monitor unit;

means for enabling counting, by the processor, of each first event associated with a primary metric of the execution of the instruction if the indicator is associated with the

instruction, wherein the processor autonomically increments the count of the first events associated with the primary metric of the execution of the instruction in a first hardware counter;

means for determining if the count of the first events associated with the primary metric of the execution of the instruction stored in the first hardware counter satisfies a predetermined relationship with a threshold value; and

means for enabling counting, by the processor, of each second event associated with a secondary metric of the execution of a portion of code associated with the instruction, wherein the processor autonomically increments the count of the second events associated with the secondary metric of the execution of a portion of code associated with the instruction in a second hardware counter.

EVIDENCE APPENDIX

There is no evidence to be presented.

RELATED PROCEEDINGS APPENDIX

There are no related proceedings.